COSC 201 – Assignment #2
Fall 2011

Objective: Create a small expression evaluator that handles basic integer arithmetic and variables.

Using the example calculator code from your book (see section 11.2 in your book) you will create a calculator that handles a variety of operations on integers, including variables with integer type. You should handle the following operations, noting that certain operations can have a different meaning when used in different settings:

Operation	Meaning
+	Unary Positive or Addition
-	Unary Negative or Subtraction
*	Multiplication
/	Integer division (i.e. 14 / 3 is equal to 4)
%	Modulo (i.e. 14 % 3 is equal to 2)
^	Power (i.e. 3^5 is equal to 243)
=	Assignment

You also will need to handle variables. To make things simpler, you should always assume that a valid variable name is a single alphabet character (a-z,A-Z) and that variable should always be initialized before use. For example:

```
> x = 5
x set to 5
> x + 4
9
```

is a good segment of code, while:

```
> x + 4
x is undefined.
```

should be handled.

You should also handle parentheses and order of operations. Aside from these characters, operators and numbers, you should assume all other characters are invalid and they should produce an error if they exist in the input line (see Testing).

Code Requirements:

You should provide the Java class to handle input from my driver (see Testing). In order to facilitate this, you must follow these specific requirements (remember, capitalization counts):

Class name: StackCalculator
Class constructor signature: public StackCalculator()
Primary Method signature: public void processInput(String s)

You will be able to test your code with my driver before needing to submit your class. If your code does not work with my driver, check your signatures with the above requirements first. Aside from the above code requirements, I have no specific implementation requirements other than your code should do the heavy lifting. No short cuts.

Testing: You should write your code with the method signatures noted above. I will be using my own driver class to run your projects and you should test your code using said driver class. The driver (Proj2Driver.java) will be posted to the course website soon and will include appropriate and informative output. The driver should not be altered by you in any way and will include the following errors that you will need to handle, printing out an informative message and exiting out of the execution of only that command (basically, your program should continue to run and get a new input even after an erroneous input). You should not use exceptions for this.

Errors:

Unbalanced parentheses.
Undefined variable.
Invalid variable name.
Invalid symbol.
Incorrect formatting.
Nonsensical input (i.e. *+2).

See above for definitions on proper variable formatting and acceptable symbols. Input will be provided to your processInput method as a single string. Note that there can be as many whitespace characters (other than newline) between characters. For instance both of the following lines are valid input:

12 + 4
12 + 4 -3

Expectations: Your code will need to be neat, concise, well documented and above all, correct (see Testing). All classes should have headers and each method should have comments describing the method's function including pre and post conditions (see www.cs.ucf.edu/~reinhard/classes/cop3503/notes01.pdf for a good introduction to pre and post conditions). Any novel or possibly confusing code should be explained, as I do get confused and distracted easily.

Grading rubric will be given out at least a week ahead of the due date. You may work in teams of up to 2 for this project. If you are looking for a teammate but have not had success locating one, you should send me an email and I'll do my best to pair you with someone.

DUE: November 9, 11:59 pm Eastern via Digital Dropbox. Submit only one set of files per team.